Tutorial 3

System Modeling with SystemC

Abhijit Ghosh, Synopsys Inc., Mountain View
Steve Tjiang, Tensilica Inc., Santa Clara
Ramesh Chandra, STMicroelectronics, San Diego

Early and accurate modeling of an entire system is a key technique in lowering the time-to-market of complex embedded SOC systems. Such systems may contain many processors and ASICs, and hundreds of thousands to millions of lines of software. Getting such systems to market fast and within specifications require early architectural exploration, software development, system integration, and system performance determination, before hardware prototypes are available. By modeling the entire system with SystemC designers can fulfill these requirements.

SystemC allow designers to model their design at multiple abstraction levels in the same environment. They can model at a relatively high abstraction level early in the design process, and later refine all or parts of the design to the style of implementation suitable for that part. This allows designers to maximize system simulation speed, while retaining the ability to model hardware components at a detailed level.

Because SystemC is based on C++, software subsystems can be specified directly in SystemC. The communication refinement features of SystemC allow designers to migrate performance sensitive parts of the software into hardware easily.

This tutorial will introduce the fundamental constructs of the SystemC libraries: the model of time, modules, processes, ports, and synchronization and communication primitives. We will discuss the more advanced features of SystemC such as channels, the separation of communication from functionality, hierarchical channels, events, static and dynamic sensitivity. Finally, we will give examples of how SystemC is used to model SOCs and how SystemC creates the framework for solving many system-level design problems. We will conclude with benefits of using SystemC to model systems as well as a roadmap of SystemC, as well as web sites and other publications for those interesting in learning more about SystemC.

System Modeling Features of SystemC

SystemC is a C++ class library that can be used to model hardware-software systems at a high level of abstraction. SystemC supports modeling pure functionality of a system, the architecture of the system, as well as hardware and software implementation details. With SystemC, you can create untimed functional models, transactional architecture models, and cycle-accurate implementation models. The following areas will be discussed

Model of Computation

Architectural exploration and early system integration require the ability to model the system at multiple abstraction levels. SystemC provides a general model of computation suitable for many levels of abstraction from software to RTL level.

0-7803-6677-8/01/$10.00©2001 IEEE.

Like other HDLs, SystemC implements a discrete time model of computation. *Processes* are the basic units of concurrent activity. Processes are grouped into *modules.* Modules can contain other modules, allowing the hierarchical construction of the system model. Processes communicate to each other via *interfaces*, *channels* and *ports*, and can synchronize with each other via *events* objects.

SystemC uses an absolute, integer-valued model of time, making it easier to combine multiple IP blocks into one simulation, by explicitly expressing the time units of each SystemC block to a common time base.

Communication Abstractions

Architectural exploration and early system integration require the ability to connect modules of different abstraction levels into a system model. SystemC provides communication abstractions to make this combination possible.

SystemC provides constructs—interfaces, ports, and channels—that allow you to build and refine the communication within your system model.

- Interfaces define a set of related communication methods (in the C++ sense). For example, an interface may define a read, write and control method to represent communication with a memory module. An interface defines an abstract type, and does not implement any functionality.

- Ports are declared as being associated with a particular interface. The port's methods call the methods of the interface. For example, if you define a memory port as using the aforementioned memory interface, the read, write, and control methods would be available for implementing the read, and write of the port.

- Channels implement one or more interface. For example, a channel that implements the aforementioned memory interface would provide implementations for the read, write and control methods. A single channel could implement one or more interfaces. When connecting two modules together, the channel could offer one interface to one module, while offering a different interface to another.

SystemC provides a set of predefined primitive channels: signals, shared variables, FIFOs, and message queues.

Communication Refinement

Architectural exploration involves refinement and decomposition of modules, and repartitioning of the model. Refinement of functionality is best accomplished in parallel with a refinement of communication. SystemC supports the latter refinement with hierarchical and composite channels.

- A hierarchical channel is a module that implements the channel. As modules are hierarchical, the channel can include other modules and channels as components This facility allows one to implement sophisticated communication protocols, or build the channel out of modules with more detailed

implementation details while providing interface at a higher-level abstraction.

- A composite channel consists of other channels as part of its interface. In the aforementioned memory interface, you could refine the read method into channels for address lines, and data lines.

Synchronization and Dynamic Sensitivity

Architectural exploration and system integration require the modeling of hardware at a behavioral model and concurrent software. SystemC provide event objects and dynamic sensitivity to facilitate this modeling. Most HDLs offer static sensitivity: processes respond to events on input signals and ports that are predetermined when you elaborate the design. In addition to static sensitivity, SystemC offers dynamic sensitivity: processes can wait explicitly wait on events that are determined at run-time.

Integration with C++

Architectural exploration, system integration and software development require that it be easy to include software into the system model. SystemC is a C++ library and users need only a C++ compiler to perform system modeling. Application software can be written directly in SystemC and added to the system model.

You can write your test benches in C++. You can write code that analyzes the performance of your model directly in C++. Such test benches and analysis code are easily integrated into your system model.